

Primeiro Programa

Boas-vindas ao guia introdutório para o seu primeiro programa Arduino! A programação com IDE Arduino é uma excelente maneira de mergulhar no mundo dos sistemas embebidos, permitindo controlar e interagir com o mundo físico através de código. Envolve a escrita de código no Ambiente de Desenvolvimento Integrado (IDE) do Arduino, onde um programa é chamado de *sketch*. Neste guia, vai aprender sobre as estruturas fundamentais de um *sketch* Arduino: as funções **setup()** e **loop()**.

A função **setup()** é onde se inicializam as definições e configurações, sendo **executada apenas uma vez** quando a placa Arduino é ligada ou reiniciada. É aqui que se definem os modos dos pinos, se inicia a comunicação serial e se realizam outras tarefas de configuração necessárias para um dado projeto.

Por outro lado, a função **loop()** contém a lógica principal do programa e é **executada continuamente**, permitindo que a placa de desenvolvimento a programar realize tarefas repetitivas, como ler sensores, controlar saídas e responder a eventos.

Ao dominar estes conceitos, estará apto a criar projetos dinâmicos e interativos que respondem ao mundo ao seu redor. Vamos começar!

O código seguinte faz com que o *Light-emitting diode* (LED) da placa de desenvolvimento *ESP32-C6-DevKitM-1* pisque, alternando entre ligado e desligado em intervalos regulares. Além disso, apresenta-se de seguida uma representação ilustrativo para referência.

```
/*
  BlinkRGB

  Demonstrates usage of onboard RGB LED on some ESP dev boards.

  Calling digitalWrite(RGB_BUILTIN, HIGH) will use hidden RGB driver.

  RGBLedWrite demonstrates control of each channel:
  void neopixelWrite(uint8_t pin, uint8_t red_val, uint8_t green_val, uint8_t blue_val)

  WARNING: After using digitalWrite to drive RGB LED it will be impossible to drive the same
  pin
  with normal HIGH/LOW level
  */
#define RGB_BRIGHTNESS 64 // Change white brightness (max 255)
```

```
// the setup function runs once when you press reset or power the board
void setup() {
  // No need to initialize the RGB LED
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(RGB_BUILTIN, HIGH); // Turn the RGB LED white
  delay(1000);
  digitalWrite(RGB_BUILTIN, LOW); // Turn the RGB LED off
  delay(1000);
}
```

[tft.gif](#)

Montagem exemplificativa

Utilização do Monitor Serial no Arduino IDE

O Monitor Serial é uma ferramenta crucial dentro do Arduino IDE que permite enviar e receber dados entre a placa de desenvolvimento e o computador. Esta funcionalidade é extremamente útil para depuração, monitorização de leituras de sensores e interação com o programa do Arduino em tempo real.

Como Funciona o Monitor Serial

O Monitor Serial comunica com a placa de desenvolvimento através de uma conexão serial, que normalmente é estabelecida através do cabo USB que liga a placa ao computador. Esta conexão permite a transmissão de dados em texto entre o seu computador e a placa de desenvolvimento. Podem enviar-se comandos do Monitor Serial para a placa de desenvolvimento, e, por sua vez, a placa de desenvolvimento pode enviar de volta dados, que são exibidos na janela do Monitor Serial.

Para utilizar o Monitor Serial, siga estes passos:

1. Inicializar a Comunicação Serial: No *sketch*, deve inicializar-se a comunicação serial utilizando a função `Serial.begin()`. Esta função recebe um único argumento, a taxa de transmissão, que define a velocidade da comunicação em bits por segundo (bps). Uma taxa de transmissão comum é 9600.

```
void setup() {
  Serial.begin(9600); // Inicia a comunicação serial a uma taxa de 9600 bps
}
```

2. Enviar Dados para o Monitor Serial: Pode enviar-se dados da placa de desenvolvimento para o Monitor Serial utilizando as funções `Serial.print()` ou `Serial.println()`. A função `Serial.print()` envia dados sem adicionar uma nova linha no final, enquanto que `Serial.println()` envia os dados seguidos de um caractere de nova linha.

```
void loop() {  
  Serial.println("Olá, Mundo!"); // Envia "Olá, Mundo!" para o Monitor serial  
  delay(1000); // Aguarda 1 segundo  
}
```

3. Receber Dados do Monitor Serial: O Monitor Serial também pode enviar dados para a placa de desenvolvimento. Podem ler-se estes dados utilizando no *sketch* as funções `Serial.read()`, `Serial.available()`, e outras funções relacionadas. No exemplo seguinte ilustra-se a leitura de um caractere usando o Monitor Serial:

```
void loop() {  
  if (Serial.available() > 0) { // Verifica se há dados disponíveis para leitura  
    char receivedChar = Serial.read(); // Lê o próximo caractere disponível  
    Serial.print("Recebido: ");  
    Serial.println(receivedChar); // Imprime o caractere recebido  
  }  
}
```

Segundo Programa

O *sketch* seguinte ilustra como modificar o *sketch* do primeiro programa anteriormente sugerido para começar a aproveitar as possibilidades do Monitor Serial:

```
/*  
  BlinkRGB  
  
  Demonstrates usage of onboard RGB LED on some ESP dev boards.  
  
  Calling digitalWrite(RGB_BUILTIN, HIGH) will use hidden RGB driver.  
  
  RGBLedWrite demonstrates control of each channel:  
  void neopixelWrite(uint8_t pin, uint8_t red_val, uint8_t green_val, uint8_t blue_val)  
  
  WARNING: After using digitalWrite to drive RGB LED it will be impossible to drive the same  
  pin  
  with normal HIGH/LOW level  
*/
```

```
//#define RGB_BRIGHTNESS 64 // Change white brightness (max 255)

// the setup function runs once when you press reset or power the board
void setup() {
  // No need to initialize the RGB LED
  Serial.begin(9600);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(RGB_BUILTIN, HIGH); // Turn the RGB LED white
  Serial.println("LED ligado"); // Imprime o caractere recebido
  delay(1000);
  digitalWrite(RGB_BUILTIN, LOW); // Turn the RGB LED off
  Serial.println("LED desligado"); // Imprime o caractere recebido
  delay(1000);
}
```

Depois garantir a seleção do dispositivo ESP32 adequado (*ESP32C6 Dev Module*), copiar o código anterior, compilar e carregar o código para a placa ESP32, deverá ser possível verificar o resultado.

tft.gif	tft.gif
-------------------------	-------------------------

Revision #27

Created 2024-08-27 11:09:01 UTC by João Pedro Monteiro

Updated 2024-09-06 09:30:24 UTC by João Pedro Monteiro